НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ В.О. Романкевич
(підпис)

"___"_____2020р.

Дипломний проєкт

освітньо-кваліфікаційного рівня "Бакалавр"

з напряму підготовки **123 "Комп'ютерна інженерія"**

на тему: «ГЕНЕРАЦІЯ ТА ВИКОРИСТАННЯ КРИПТОГРАФІЧНОГО КЛЮЧА ДЛЯ СМАРТ-КАРТ»

Виконав: студент 4 курсу, групи КВ-62

Війтенко Артем Максимович _____ (підпис)

Керівник асистент каф. СПіСКС Радченко К.О. _____ (підпис)

Консультант з нормоконтролю доц., к.т.н. Клятченко Я.М. _____ (підпис)

Рецензент . _____ (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2020

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 123 «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Романкевич В.О.

(підпис) (ініціали, прізвище)

«___» ч_____2020 р.

ЗАВДАННЯ

на дипломний проект студента

Війтенка Артема Максимовича
(прізвище, ім’я, по батькові)

1. Тема проекту

«Генерація та використання криптографічного ключа для смарт-карт», керівник проекту Радченко Костянтин Олександрович, асистент каф. СПіСКС,, затверджені наказом по університету від «25» травня 2020 р. №1181-С

2. Термін ~~подання~~ студентом проекту «_ » червня 2020 р.

3. Вихідні дані до проекту див. Технічне завдання

4. Зміст пояснювальної записки

- Аналіз існуючих рішень
- Методи та технології
- Додаток

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

- Алгоритм шифрування. Схема алгоритму
- Схема дешифрування файлу. Схема алгоритму
- Мікропроцесорні смарткарти. Схема структурна
- Алгоритм шифрування AES(Rijndael). Схема алгоритму

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М. доцент		

7. Дата видачі завдання «_ » _____2020 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Вивчення літератури за тематикою проекту	17.11.2019	
2	Розроблення та узгодження технічного завдання	8.02.2020	
3	Аналіз існуючих рішень	12.02.2020	
4	Підготовка матеріалів першого розділу дипломного проєкту	25.02.2020	
5	Підготовка матеріалів другого розділу дипломного проєкту	15.03.2020	
6	Розроблення програмного забезпечення	4.04.2020	
7	Відлагодження програмного продукту	10.04.2020	
8	Підготовка матеріалів третього розділу дипломного проєкту	24.04.2020	
9	Підготовка графічної частини дипломного проекту	19.05.2020	
10	Оформлення документації дипломного проекту	26.05.2020	

Студент _____ Війтенко А.М.

Керівник проекту _____ Радченко К.

АННОТАЦІЯ

Мета дипломного проєкту дослідити і структурувати теоретичні відомості необхідні для реалізації криптографічного шифрування даних за допомогою технології «смарт-карт» і на основі отриманих знань розробити власний додаток.

Для реалізації описаної мети проведено аналіз існуючих файлових систем та додатків, за допомогою яких можна шифрувати дані, алгоритмів криптографічного шифрування та технології «смарт-карт».

В результаті роботи був програмно реалізований додаток, у якому було використано оптимальний алгоритм шифрування у поєднані з фізичним носієм – смарт-картою. Перевагою даного проєкту є висока захищеність від кейлоггерів та інших програм злому.

Результати дипломної роботи можуть бути використанні для вивчення основних концепцій шифрування даних та технології «смарт-карт».

Дипломний проєкт містить: 50 ст., 17 рис., 12 посилань на використані джерела.

Ключові слова: смарт-карт, криптографічне шифрування, алгоритм rijndael, криптографічний ключ, кейлоггер

ABSTRACT

The aim of the graduation project is to research and structure the theoretical information necessary for the implementation of cryptographic data encryption using the technology of "smart cards" and based on the knowledge gained, develop new application.

To implement the described goal, an analysis was made of existing file systems and applications with which you can cipher data, cryptographic encryption algorithms and smart card technology.

As a result of the work, an application was implemented in software, in which the optimal encryption algorithm was used in combination with a physical medium - a smart card. The advantage of this project is its high security against keyloggers and other hacking programs.

The results of the thesis can be used to study the basic concepts of data encryption and smart card technology. The graduation project contains: 50 pages, 17 figures, 12 references to the sources used.

Keywords: smart cards, cryptographic encryption, rijndael algorithm, cryptographic key, keylogger.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ. 045470.002 ТЗ	Генерація та використання	4		
			криптографічного ключа			
			для смарт-карт			
			Технічне завдання			
	А4	ІАЛЦ.045470.003 ТП	Генерація та використання	1		
			криптографічного ключа			
			для смарт-карт			
			Відомість технічного			
			проєкту			
	А4	ІАЛЦ.045440.004 ПЗ	Генерація та використання	50		
			криптографічного ключа			
			для смарт-карт			
			Пояснювальна записка			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045470.001 ОА		
Розробив		Війтенко А.М.			Генерація та використання криптографічного ключа для смарт-карт ***Опис альбому***	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Радченко К.О.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-62	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість	№	Примітки
	А4	ІАЛЦ.045470.005 ДЗ	Алгоритм шифрування.	1		
			Схема алгоритму			
	А4	ІАЛЦ.045470.006 ДЗ	Схема дешифрування	1		
			Схема алгоритму			
	А4	ІАЛЦ.045470.007 ДЗ	Мікропроцесорні	1		
			смарт-карти			
			Схема структурна			
	А4	ІАЛЦ.045470.008 ДЗ	Алгоритм шифрування	1		
			AES(Rijndael).			
			Схема алгоритму			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

					ІАЛЦ.045470.001 ОА	Арк.
						2
Змін.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

					ІАЛЦ. 45470.002 ТЗ		
Змін	Арк.	№ докум.	Підпис	Дата			
Розробив		Війтенко А.М.			Генерація та використання криптографічного ключа для смарт-карт ***Технічне завдання***	Літ.	Аркуш 1 / Аркушів 4
Перевірив		Радченко К.О.					
Н. контроль		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ФПМ КВ-62	
Затвердив		Романкевич В.О					

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Генерація та використання криптографічного ключа для смарт-карт».

Галузь застосування: захист інформації.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3 МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є розробка власного додатку для забезпечення захисту даних, використовуючи криптографічне шифрування і технологію «смарт-карт».

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5 ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється

- Сумісність з операційною системою Windows;
- Генерація криптографічного ключа;
- Шифрування данних;
- Зберігання криптографічного ключа на фізичному носії – смарт-карті;
- Наявність зручного меню;

5.2 Вимоги до апаратного забезпечення

- Процесор: Intel Core i5-6200;
- Оперативна пам’ять: 8 Гб;

5.3 Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Windows;
- Зчитувач для безконтактних смарт-карт ACR1222L;
- Драйвер для зчитувача безконтактних смарт-карт ACR1222L;
- Смар-карта MIFARE 2K.

ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	10.02.2020
2.	Розроблення та узгодження технічного завдання	15.02.2020
3.	Аналіз існуючих рішень	20.04.2020
4.	Підготовка матеріалів розділів дипломного проекту	25.04.2020
5.	Підготовка звіту дипломного проекту	10.05.2020
6.	Передзахист дипломного проекту	20.05.2020

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість	№ прим.	Примітки
	А4	ІАЛЦ. 045470.002 ПЗ	Генерація та використання	50		
			криптографічного ключа			
			для смарт-карт			
			Пояснювальна записка			
	А4	ІАЛЦ.045470.005 Д3	Алгоритм шифрування.	1		
			Схема алгоритму			
	А4	ІАЛЦ.045470.006 Д3	Схема дешифрування.	1		
			Схема алгоритму			
	А4	ІАЛЦ.045470.007 Д3	Мікропроцесорні смарт-карти	1		
			Схема структурна			
	А4	ІАЛЦ.045470.008 Д3	Алгоритм шифрування Rijndael(A	1		
			Схема алгоритму			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.003 ТП		
Розробив		Війтенко А.М.			Генерація та використання криптографічного ключа для смарт-карт ***Відомість технічного проекту***	Літ.	Аркуш 1 / Аркші 1
Перевірила		Радченко К.О.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-62	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

CGD - Драйвер CryptoGraphic Disk.

NetBSD - операційна система.

PKCS - Public Key Cryptography Standards (Стандарти криптографії з відкритим ключем).

SFS – драйвер Secured File System.

EFS - Encrypting File System, файлова система шифрування даних.

ОС – операційна система.

ВСТУП

Криптографія це невід'ємна частина життя кожної людини нашого часу, ця наука звертає на себе дуже багато уваги. Досягнення у цій галузі визначають захищеність наших особистих даних та особистого життя. У зв’язку з цим удосконалення цієї системи має велике значення для усіх нас.

Актуальність теми дипломної роботи полягає в тому, що майже до будь-якої інформації на вашому комп'ютері можна отримати доступ, але якщо поєднати технологію криптографічного шифрування та смарт-карту, можна отримати надійно захищенні данні.

Предмет дослідження – використання криптографічного ключа для смарт-карт, щоб забезпечити захист певних даних.

Об’єкт дослідження – поєднання криптографічного ключа з технологією «смарт-карт».

Мета дипломного проєкту: розробка власного додатку для забезпечення захисту даних.

Для досягнення мети дипломного проєкту поставлено такі завдання :

- Аналіз існуючих додатків та файлових систем які реалізують захист інформації.
- Долідження методів шифрування.
- Дослідження технології «смарт-карт».
- Дослідження технології зчитувачів для смарт-карт.

Теоретичною основою для дослідження стали книжки та статті зарубіжних авторів а також офіційні документації певних додатків.

Дипломний проєкт складається з: вступу, трьох розділів, що включають 8 підрозділів, висновків, списку використаних джерел із N найменувань. У тексті дипломної роботи міститься 17 рисунків. Загальний обсяг роботи 51 сторінка.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1. Проблема, яку вирішує продукт

Хочеться на самому початку сказати, що будь-які зашифровані файли не можуть бути захищенні на всі 100%, зловмисники можуть обійти захист, хоч із труднощами. Зберігання криптографічних ключів і паролів в незашифрованому вигляді, а також встановлені без відома користувача за допомогою шкідливого ПО кейлоггери (keyloggers), становлять загрозу комп'ютерної безпеки. У нашому випадку ключ буде знаходитись на фізичному носії – смарт-карті, що значно зменшить ризик втрати даних.

1.2. Аналоги

Цей підрозділ описує низку доступних методів шифрування даних – системи шифрування файлів, які організовуються за рахунок підвищення рівня абстрагування: блокові системи, файлові системи на основі диска, мережеві системи на основі циклу, файлові системи, що складаються та програми. Кожен метод розглянемо більш детально з прикладами та зробимо висновок.

Блокові системи: системи шифрування на основі блоків працюють нижче рівня файлової системи, шифруючи один блок за один раз. Це вигідно, оскільки вони не потребують знань про файлову систему, яка знаходиться над ними, і навіть можуть використовуватися для підкачування розділів або додатків, які потребують доступу до необроблених розділів (наприклад, серверів баз даних). Крім того, вони не розкривають інформацію про окремі файли (наприклад, розміри та власника) або структуру каталогів. Прикладами блокових систем є Cryptoloop, Cryptographic Disk, GEOM Based Disk Encryption, Secured File System.

Cryptoloop - драйвер пристрою петлевого звороту Linux представляє файл у вигляді блочного пристрою, необов'язково перетворюючи дані до їх запису та після його зчитування з власного файлу, як правило, для забезпечення шифрування. Ядра Linux містять криптографічну основу, CryptoAPI, яка експортує єдиний інтерфейс для всіх шифрів і хешів. В даний час IPsec та драйвер Cryptoloop використовують ці засоби.

CGD - Драйвер, доступний в NetBSD, схожий з пристроєм для зворотного зв'язку Linux та іншими системами шифрування пристрою циклу, але він використовує власний диск або розділ як сховище резервного копіювання. CGD має повнофункціональний утиліти конфігурації простору користувача, які включають n-фактор аутентифікації та PKCS №5 для перетворення паролів користувачів у ключі шифрування. Ця система схожа на Cryptoloop, використовуючи необроблений пристрій в якості резервного магазину. Має декілька недоліків:

- Перший - він не може захистити дані, коли ви обмінюєтесь даними між пристроями або надсилаєте дані електронною поштою, дані в передачі не захищені.
- Другий - у випадку, коли весь диск шифрується, кожного разу, коли ви намагаєтесь прочитати дані накопичувача, від ключа аутентифікації вимагається розшифрувати дані диска. Цей процес сповільнить ваш комп'ютер.
- Третій - повне шифрування диска може ускладнювати відновлення даних на диску. Якщо зашифровані дані можно було легко відновити, шифрування не мало б сенсу. Отже, відновлення даних, безумовно, важко.

Шифрування дисків GBDE базується на GEOM, який забезпечує модульну основу для виконання перетворень, починаючи від простого геометричного переміщення для розділення диска, до криптографічного захисту збережених даних. GBDE - перетворення GEOM, яке дозволяє

шифрувати весь диск. GBDE хеширує додану користувачем парольну фразу на 512 біт ключового матеріалу. GBDE використовує основний матеріал для пошуку та шифрування головного ключа 2048 біт та інших метаданих у чотирьох різних секторах блокування . Коли шифрується окремий сектор, номер сектору та біти головного ключа об'єднуються за допомогою MD5 для створення ключа . Випадково генерований ключ, секторний ключ, зашифрований ключем, а потім записується на диск. Нарешті, корисне навантаження сектора шифрується за допомогою секторного ключа та записується на диск. Ця техніка, хоча і більш складна, схожа на Cryptoloop, використовуючи необроблений пристрій в якості резервного магазину. На відміну від громіздких систем шифрування, які шифрують окремі файли, gbde шифрують в прозорому режимі файлову систему в цілому, при цьому дані у відкритому вигляді на диск ніколи не записуються.

SFS - це драйвер пристрою MSDOS, який шифрує весь розділ . SFS схожий на Cryptoloop, використовуючи необроблений пристрій як резервний магазин. Після шифрування драйвер представляє розшифрований вигляд зашифрованих даних. Це забезпечує зручну абстракцію файлової системи, але покладатися на MSDOS ризиковано, оскільки MSDOS не забезпечує жодного із захистів сучасної операційної системи [1].

BestCrypt - це комерційно доступний драйвер пристрою для зворотного зв'язку, який підтримує багато шифрів. BestCrypt підтримує і Linux, і Windows, і використовує звичайний файл в якості резервного сховища (подібно до використання попередньо виділеного файлу з Cryptoloop) [2].

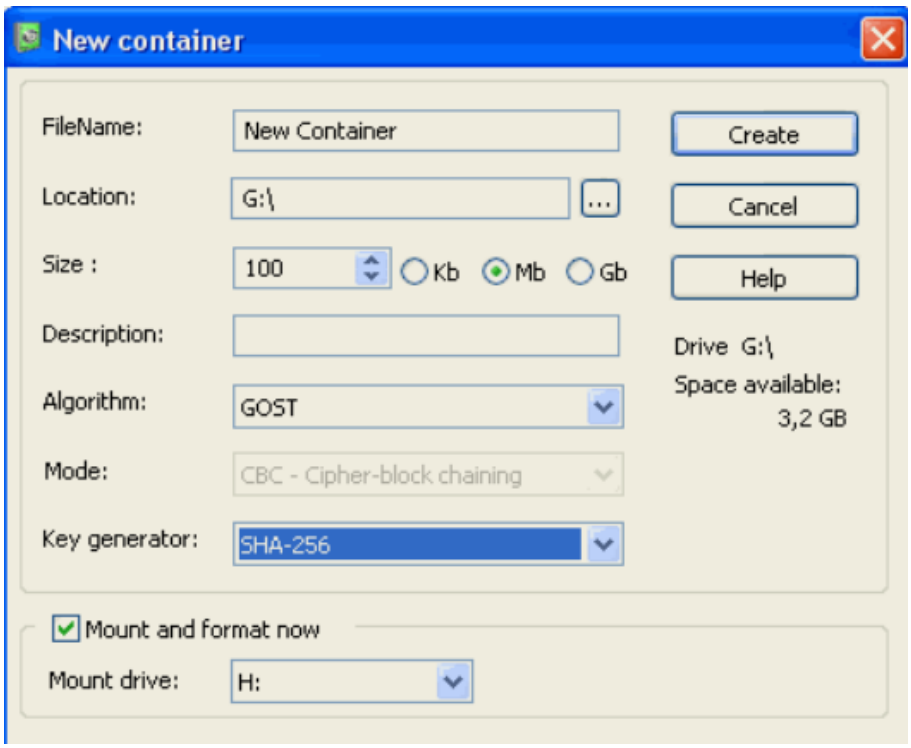

Рис. 1 створення зашифрованого диску в BestCrypt.

Створює на жорсткому диску компю'ютера віртуальний зашифрований диск (контейнер), при створені можна вибрати алгоритм шифрування, розмір контейнеру та багато іншого (Рис. 1). Він працює, як звичайний дисковий розділ. Шифрування і розшифрування йдуть у фоновому режимі, і користувач не помічає різниці в роботі зі звичайним і зашифрованим диском, який, при необхідності, можна перетворити в звичайний, але нечитаний, файл. Має такі додаткові функції:

- Можливість створення прихованого контейнера в уже створеному явному контейнері;
- можливість зберігання контейнерів на будь-яких типах носіїв (як мережевих, так і локальних) з можливістю їх переміщення, копіювання, дублювання зі збереженням всіх захисних можливостей;
- можливість закриття контейнера по "гарячим" клавішах або після певного часу при неактивності користувача;
- можливість шифрування свап-файлу;
- функція захисту контейнера від випадкового видалення; утиліта гарантованого знищення даних.

Незважаючи на багато переваг, BestCrypt має свої недоліки. Програма забезпечує надійний захист конфіденційних даних, однак, файл, в якому зберігається інформація про контейнері, можна легко видалити. Файл з даними легко видаляється під іншою системою або навіть в тій же системі, якщо BestCrypt не працює. Другим істотним недоліком програми є те, що запустивши, її вже неможливо вивантажити без втручання Task Manager. Таким чином, у нижній частині екрана постійно знаходиться ікона, яка повідомляє про те, що вам є що приховувати від оточуючих.

Файлові системи на основі диска: дискові файлові системи, що шифрують дані, розташовані на більш високому рівні абстракції, ніж системи на основі блоків. Ці файлові системи мають доступ до всіх даних по файлу та по каталогу, тому вони можуть виконувати більш складні авторизацію та аутентифікацію, ніж системи, засновані на блоках, але в той же час файлові системи на основі диска можуть керувати фізичною компонуванням даних. Це означає, що файлові системи на основі диска можуть обмежувати кількість інформації, виявленої зловмиснику, про розмір файлу та власника, хоча на практиці ці атрибути часто все-таки розкриваються з метою збереження структури дискової системи файлової системи. Крім того, оскільки немає додаткового шару непрямості, файлові системи на основі диска можуть мати переваги в порівнянні з іншими методами, описаними в цьому розділі (включаючи петлеві пристрої).

EFS - це файлова система шифрування в Microsoft Windows на основі ядра NT (Windows 2000 та XP). Це розширення до NTFS та використовує методи аутентифікації Windows, а також Windows ACL. Хоча EFS розташований у ядрі, він щільно поєднується з DLL-кодами простору користувача для виконання шифрування та локальним сервером аутентифікації безпеки для простору користувача. Це запобігає використанню EFS для захисту файлів або папок у кореневому або \ winnt каталозі.

Ключі шифрування зберігаються на диску в блоці який шифрується за допомогою пароля для входу користувача. Це означає, що коли користувачі змінюють свій пароль, блокування блоків має бути зашифровано повторно. Якщо адміністратор змінить пароль користувача, то всі зашифровані файли стають нечитабельними. Крім того, для сумісності з Windows 2000 EFS використовує DESX за замовчуванням, і єдиним іншим доступним шифром є 3DES (включений у Windows XP або в пакеті Windows 2000 High Encryption). Основним недоліком є те, що при використанні цієї файлової системи, незашифрована версія зберігається в тимчасовій пам'яті, так що зловмисник може мати до неї доступ [3].

StegFS - це файлова система, яка використовує стеганографію, а також шифрування . Якщо взломисники перевіряють систему, вони знають лише, що є деякі приховані дані. Вони не знають змісту чи обсягу того, що приховано. Це досягається за допомогою модифікованого драйвера ядра Ext2, який зберігає окрему таблицю розподілу блоків на рівень безпеки. Неможливо визначити, скільки рівнів безпеки існує без ключа для кожного рівня безпеки. Коли диск встановлений з немодифікованим драйвером Ext2, випадкові блоки можуть бути перезаписані, тому дані реплікуються випадковим чином на весь диск, щоб уникнути втрати даних. Хоча StegFS досягає правдоподібної заперечуваності існування даних, погіршення продуктивності становить коефіцієнт 6-196, що робить непрактичним для більшості застосувань.

Мережеві системи на основі циклу: мережеві файлові системи (NBFS) працюють на більш високому рівні абстракції, ніж файлові системи на основі диска, тому NBFS не можуть контролювати макет файлів на диску. NBFS мають дві основні переваги: по-перше вони можуть працювати над будь-якою файловою системою, і по-друге вони більш портативні, ніж файлові системи на основі диска.

Основними недоліками NBFS є ефективність та безпека. Оскільки кожен запит повинен подорожувати через мережевий стек, потрібно більше копій даних і продуктивність погіршується. Безпека також страждає, оскільки NBFS вразливі до всіх слабких сторін базового мережевого протоколу (як правило, NFS).

CFS - це криптографічна файлова система, реалізована як сервер NFS рівня користувача. Шифр і ключ задаються при першому створенні зашифрованих каталогів. CFS демон відповідає за надання власника , що має доступ до зашифрованих даних з допомогою *приєднувати* командування. Демон – комп'ютерна програма в системних класах UNIX, використовуючи самої системи і працює в фоновому режимі без прямого взаємодії з користувачем. Демони зазвичай використовуються під час завантаження систем. Демон, після перевірки ідентифікатора користувача та ключа, створює в каталозі точки монтування каталог, який виконує роль незашифрованого вікна зашифрованих даних користувача. Після додавання користувач отримує доступ до доданого каталогу, як і будь-який інший каталог. CFS - ретельно розроблена портативна файлова система з широким вибором вбудованих шифрів. Однак його основна проблема - продуктивність. Оскільки він працює в режимі користувача, він повинен виконувати багато контекстних комутаторів і копій даних між ядром і користувальницьким простором. Оскільки CFS має немодифікований клієнт NFS, який спілкується зі зміненим сервером NFS, він повинен працювати лише через мережевий інтерфейс зворотного зв'язку.

TCFS - це криптографічна файлова система, реалізована як модифікований клієнт NFS-режиму ядра. Оскільки він використовується спільно з сервером NFS, TCFS прозоро працює з віддаленою файловою системою, усуваючи необхідність у конкретних командах приєднання та від'єднання. Для шифрування даних користувач встановлює зашифрований

атрибут на каталоги та файли в точці монтажу NFS. TCFS інтегрується із системою аутентифікації UNIX замість того, щоб вимагати окремих

парольних фраз. Він використовує базу даних у `/ etc / tcfspwdb` для зберігання зашифрованих ключів користувачів та груп. Груповий доступ до зашифрованих ресурсів обмежений підгруппою членів даної групи UNIX, одночасно допускаючи механізм (так званий пороговий таємний обмін) для реконструкції групового ключа, коли члена групи більше немає. TCFS використовує модифікований клієнт NFS, який повинен бути реалізований у ядрі. Однак це дозволяє йому працювати над будь-яким мережевим інтерфейсом та працювати з віддаленими серверами. TCFS має ряд слабких місць, які роблять його менш корисним для розгортання. По-перше, посилання на паролі для входу як користувацькі ключі не є безпечною. Також зберігання ключів шифрування на диску в базі даних ключів ще більше знижує безпеку. Нарешті, TCFS доступний лише в системах з ядром 2.2.17 Linux або новіших версій, що обмежує його доступність.

Файлові системи, що складаються: це компроміс між файловими системами на основі ядра та мережевими файловими системами. Файлові системи, які можна складати, можуть працювати над будь-якою файловою системою; їм не потрібно копіювати дані через межу ядра користувача або через мережевий стек; і вони портативні для декількох операційних систем.

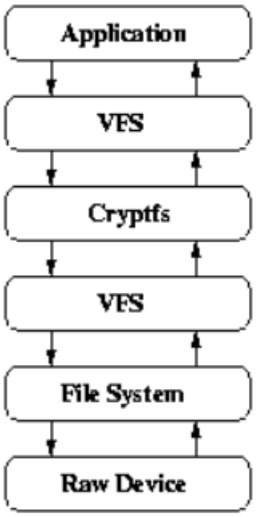

Рис. 2 функціонування Cryptfs.

Cryptfs - це криптовалютна файлова система, що складається з даних, і є частиною набору інструментів FiST . Cryptfs ніколи не був розроблений як захищена файлова система, а скоріше доказ концепції застосування FiST . Cryptfs підтримує лише один шифр і реалізує обмежену схему управління ключами. Cryptfs служить основою для декількох комерційних та дослідницьких систем (наприклад, ZIA). Користувацька програма викликає системний виклик через віртуальну файлову систему (VFS). VFS викликає файлову систему, що зберігається, яка знову викликає VFS після шифрування або розшифрування даних. VFS викликає файлову систему нижчого рівня, яка записує дані у сховище резервного копіювання(Рис. 2).

NCryptfs - це криптографічна файлова система, що може складатись, створена з явною метою збалансування безпеки, зручності та продуктивності. NCryptfs дозволяє системним адміністраторам та користувачам налаштувати NCryptfs відповідно до їх конкретних потреб. NCryptfs підтримує декілька паралельних методів аутентифікації, декілька динамічно завантажуваних шифрів, спеціальних груп та аутентифікації у відповідь на виклик. Ключі, активні сеанси та авторизація в NCryptfs у всіх мають тайм-аути.

NCryptfs можна налаштувати на прозоре призупинення та відновлення процесів на основі ключа, сеансу чи дійсності авторизації. NCryptfs також покращив ядро для відмови від сторінок прозорого тексту та сповіщення файлової системи про вихід процесу, щоб викреслити недійсні записи аутентифікації. NCryptfs підтримує нативні групи UNIX, як і будь-яке інше об'єднання. Група UNIX має деякі недоліки, перш за все, тим, що групу потрібно налаштувати системний адміністратор достроково. Це означає, що користувачі повинні звернутися до системного адміністратора, а потім чекати, коли будуть вжиті дії. NCryptfs підтримує спеціальні групи, просто додаючи авторизацію для декількох окремих користувачів (або інших об'єктів). Проблема такого підходу полягає в тому, що кожен додатковий користувач повинен мати дозволи для зміни об'єктів нижчого рівня, оскільки NCryptfs за замовчуванням дотримується стандартних перевірок файлової системи нижнього рівня. Якщо дозволи на об'єкти нижчого рівня послаблені, нові користувачі можуть змінювати файли. Однак без відповідної групи UNIX важко дати дозволи саме підгрупі користувачів, які мають їх мати. Якщо дозволи занадто розслаблені, користувачі-шахраї можуть тривіально знищити дані, і криптоаналіз стане простішим - навіть без того, щоб система була порушена.

Програми: шифрування файлів може виконуватися різноманітними додатками, що знаходяться над файловою системою. На сьогоднішній день створено дуже багато додатків для шифрування даних, далі ми розглянемо більш детально деякі популярні додатки для шифрування: GNU Privacy Guard, veracrypt та axcrypt.

GnuPG використовує шифрування з відкритим ключем, щоб користувачі могли безпечно спілкуватися. Ця програма може використовуватись для симетричного шифрування, але основна програма використовується для асиметричного шифрування інформації.

При ассиметричному шифровані використовуються 2 ключа - публічний і приватний [4]. Публічний використовується для шифрування і може бути наданий кожному, з ким користувач хоче спілкуватися, а приватний - для розшифровки, і його потрібно зберігати в безпеці. Завдяки такій схемі розшифрувати повідомлення може лише власник приватного ключа (навіть тот, хто зашифровував повідомлення, не може створити зворотну операцію). GnuPG використовує дещо складнішу схему, в якій користувач має первинний ключ, а потім нуль або більше додаткових підпорядкованих ключів. Первинна і підпорядкована панелі ключів поєднуються для полегшення управління ключами, і пакет часто може розглядатися просто як один ключ.

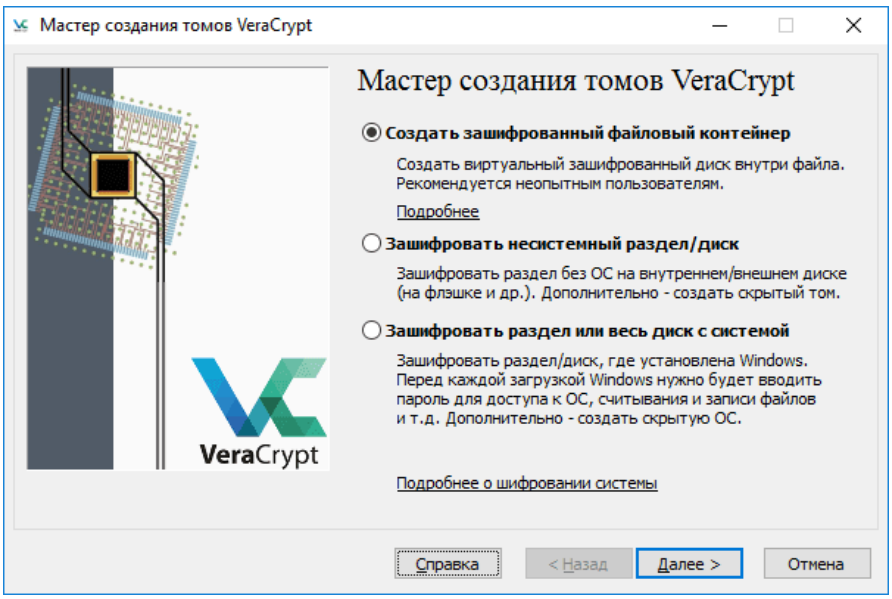

Рис. 3 мастер шифрування VeraCrypt

VeraCrypt - це безкоштовний програмний інструмент для шифрування з відкритим кодом. Він орієнтований на локальне шифрування. За допомогою нього можна зашифрувати локальні файли та папки, а також цілі жорсткі диски (Рис. 3). Програма має кілька алгоритмів шифрування, таких як AES, Serpent, Twofish, Camellia, а також комбінації цих алгорітмов. та хешування (ripemd-160 sha-256 sha-512 whirlpool streebog), що дозволяє налаштувати захист. Оскільки це локальне рішення, у VeraCrypt менше ризику, ніж у хмарній службі шифрування [5].

Програма має великий рівень надійності захисту інформації але має певні недоліки:

- Ключі VeraCrypt зберігаються у відкритому вигляді в пам'яті і при фізичному допуск до комп'ютера і при наявності відповідної програми, ключ можна вкрасти. VeraCrypt не може запобігти кешованих паролів, ключів шифрування, а також вміст конфіденційних файлів, відкритих в ОЗП можуть бути не збережені в незашифрованому вигляді в файли дампа пам'яті.
- Дефрагментація. Не дефрагментувати файлові системи, в якій зберігаються обсяги VeraCrypt.
- хеш-функція Streebog погано працює з алгоритмом шифрування "Кузнечик".

AxCrypt - це програмне забезпечення для шифрування з відкритим кодом, що пропонує безкоштовну і платну версіъ для Microsoft Windows , macOS , Android та iOS . Він може стискати, видаляти, шифрувати / розшифровувати та редагувати файли (хоча поки що версії Android, iOS та безкоштовні програми macOS - це програми лише для перегляду та читання). Це файлова програма шифрування, яка відрізняється від програми шифрування на основі контейнера, тобто кожен окремий захищений файл шифрується програмою індивідуально, а не програма, що містить усі захищені файли в один зашифрований контейнер, як TrueCrypt або VeraCrypt . AxCrypt може зашифрувати файл або папку, використовуючи або файл ключа, або пароль. Ця програма також може захищати файли в службах спільного використання файлів, таких як DropBox або Google Drive. Використовує стандарт шифрування AES-128 та AES-256. Користувачі можуть увійти, ввівши пароль для шифрування файлів або папок. Поки користувач увійшов у систему, він може відкривати захищені файли та папки. Файл ключа - це власне текстовий документ, що містить код для відкриття файлу [6].

Існуючий файл також може використовуватися для шифрування файлу чи папки. AxCrypt дозволяє відкрити захищені файли іншими користувачами додатку з тим самим паролем, який був використаний для шифрування. AxCrypt також може безпечно видаляти дані: інструмент розшифровує дані та потім видаляє їх. Мета цієї функції - не допустити зловмисного користувача пізніше отримувати дані.

Можна зробити такий висновок, що майже усі програмні рішення є досить незручними для користувачів. Кожен раз, коли вони хочуть отримати доступ до файлу, користувачі повинні вручну розшифрувати або зашифрувати його. Чим більше потрібно взаємодії з користувачем для шифрування або розшифрування файлів, тим частіше допускаються помилки, що призводить до пошкодження файлу або протікання конфіденційних даних. Крім того, файл може перебувати в чистому тексті на диску, поки користувач активно працює над ним. Шифрування файлів також може бути інтегровано у кожну програму (наприклад, текстові редактори чи поштові клієнти), але це перекладає навантаження з користувачів на програмістів. Часто розробники додатків не вважають, що додаткові зусилля щодо реалізації функцій виправдані, коли лише незначна частина користувачів скористається цими функціями. Навіть якщо шифрування вважається достатньо важливою функцією, яку можна інтегрувати в більшість додатків, все ж існують дві основні проблеми з таким підходом. По-перше, кожний додатковий додаток, якому користувач повинен довіряти, щоб правильно функціонувати, знижує загальну безпеку системи. По-друге, оскільки кожна програма може реалізувати шифрування трохи по-іншому, це ускладнить використання файлів в окремих програмах.

1.3. Сфери застосування

Розроблений додаток може бути використан як і у повсягденному житті так і сферах де необхідна надійна захищеність даних. На відміну від інших додатків, наш має додаткові перешкоди для взлому, не рахуючи самого шифру: наявність зчитувача для безконтактних смарт-карт, знання про конкретний вид смарт-карти та можливість поставити додатковий захист на смарт-карті від взлому. Навіть якщо зловмисник заволодіє фізичним носієм(смарт-картою), знадобиться дуже багато часу, щоб отримати доступ до інформації на цій карті.

2. МЕТОДИ ТА ТЕХНОЛОГІЇ

2.1. Алгоритм методу шифрування

Шифри: для криптографічних файлових систем може використовуватися кілька шифрів, як правило, є симетричними блоковими шифрами. Це тому, що блок-шифри ефективні та універсальні. Ми обговорюємо варіанти DES, Blowfish та Rijndael, оскільки вони часто використовуються для шифрування файлів і вважаються надійними. Є багато інших блок-шифрів, серед яких CAST, GOST, IDEA, MARS, Змій, RC5, RC6 та TwoFish. Більшість з них мають схожі характеристики з різними розмірами блоків та ключів.

DES - це блок-шифр, розроблений IBM за сприяння АНБ у 1970-х роках. DES був першим алгоритмом шифрування, який був опублікований як стандарт NIST з достатньою кількістю деталей, щоб їх реалізувати в програмному забезпеченні. DES використовує 56-бітний ключ, розмір 64-бітного блоку, і його можна ефективно реалізовувати в апараті. DES більше не вважається безпечним. Є кілька більш безпечних варіантів DES, найчастіше 3DES. 3DES використовує три окремі шифрування DES з трьома різними ключами, збільшуючи загальну довжину ключа до 168 біт. 3DES вважається безпечним для урядових комунікацій. DESX - це варіант, розроблений RSA Data Security, який використовує другий 64-бітний ключ для відбілювання (затемнення) даних до першого раунду та після останнього раунду DES, тим самим зменшуючи його вразливість до нападів грубої сили, а також диференціального та лінійного криптоаналізу [7].

Blowfish - це блок-шифр, розроблений Брюсом Шнейєром із 64-бітовим розміром блоку та розмірами ключів до 448 біт. Blowfish мав чотири дизайнерські критерії: швидкість, компактне використання пам'яті, прості операції та змінна безпека. Blowfish найкраще працює, коли ключ не змінюється часто, як це відбувається у випадку шифрування файлів, оскільки

підпрограми настройки ключа вимагають 521 ітерації шифрування Blowfish. Blowfish широко використовується для шифрування файлів [7].

AES (Rijndael) є спадкоємцем DES, обраного на публічному конкурсі. Хоча всіх шести фіналістів було визнано достатньо безпечними для AES, остаточним вибором для AES став Ріндделйл, заснований на складі трьох критеріїв відбору (безпека, вартість та характеристики алгоритму). Rijndael - це блоковий шифр на основі шифру Square, який використовує S-коробки (підміна), перенесення та XOR для шифрування 128-бітних блоків даних. Rijndael підтримує 128, 192 та 256 бітові ключі (далі цей алгоритм буде розглянуто більш детально).

Для данного проєкту було вирішено використати саме метод шифрування - Rijndael, тому що має найкраще поєднання безпеки, продуктивності, ефективності, простоти реалізації та гнучкості.

Rijndael був розроблений бельгійськими криптографами Джоан Дамен з International Proton World International та Вінсента Ріджмена з Kathlieke Universiteit Leuven . Розроблений ними алгоритм був зроблений як легка зрозуміла математична структура, яку можна розбити на прості компоненти. Daemen і Rijmen пишуть у своїй пропозиції до AES, що Rijndael був розроблений на основі трьох таких критеріїв :

- Опір проти всіх відомих атак;
- Швидкість та компактність коду на широкому діапазоні платформ;
- Простота дизайну

Rijndael - алгоритм блочного симетричного шифрування, який у 2002 році вибрав Національний інститут науки і техніки (NIST) як розширений стандарт шифрування (AES). Він перевершує стандарт шифрування даних (DES). NIST вибрав Rijndael як стандартний алгоритм шифрування симетричного ключа, який буде використовуватися для шифрування

некласифікованої федеральної інформації. Вибір ґрунтувався на ретельному та всебічному аналізі характеристик безпеки та ефективності алгоритму Rijndael.

Rijndael - ітераційний блок-шифр, шифрування або дешифрування блоку даних здійснюється шляхом ітерації (раунду) конкретного перетворення (раундова функція). Як вхід, Rijndael приймає одновимірні 8-бітові байтові масиви, які створюють блоки даних. Пласт-текст вводиться, а потім відображається в байтах стану. Ключ шифру - це також одновимірний 8-бітовий байтовий масив. З ітераційним блоковим шифром різні перетворення діють послідовно на проміжних результатах шифрів (станах). Конструкція Rijndael базується на легко зрозумілих математичних концепціях, включаючи математику з кінцевими полями та лінійну алгебру для матричного маніпулювання.

Основна особливість цього алгоритму це його здатність працювати на різних розмірах ключів і блоків даних. Це забезпечує додаткову гнучкість, оскільки розмір ключа і розмір блоку можуть становити 128, 192 або 256 біт. Оскільки алгоритм визначає три розміри клавіш, це означає, що існує приблизно 3,4 x 1038 можливих 128-бітних ключів, 6,2 x 1057 можливих 192-бітних ключів і 1,1 x 1077 можливих 256-бітних ключів. Для порівняння, DES-ключі мають лише 56 біт, а це означає, що існує DES 7,2 x 1016 можливих ключів DES.

Підключі виводяться з ключа шифру, використовуючи розклад клавіш Rijndael. Ключ шифру розширюється для створення розгорнутого ключа, а підпункт створюється шляхом отримання «раундового ключа» за допомогою раундового ключа. Необхідна довжина раундового ключа дорівнює довжині блоку даних, помноженій на кількість раундів плюс 1. Отже, раундові ключі беруть із розгорнутого ключа. Для підтримки захищеної системи розширений ключ завжди виводиться з ключа шифру. Цей метод гарантує, що розширений ключ ніколи не буде вказаний безпосередньо, що відкриє Rijndael до декількох

криптоаналітичних атак проти його методів генерації ключів. Нагадаємо, що безпека цієї системи повністю залежить від секретності ключа, оскільки сама конструкція алгоритму є загальнодоступною і не містить секретності.

Операції з цілим байтом: існує кілька математичних попередніх записів, які визначають операції додавання та множення в кінцевому полі та з матрицями. Виконуючи кінцеву математику, байти розглядаються як поліноми, а не числа, що може дозволяти різним, а іноді і допускає більш прості реалізації.

Шифр Rijndael - це інтерактивний блок-шифр. Тому він складається з послідовності перетворень для шифрування або розшифровки даних. Шифрування та дешифрування Rijndael починаються та закінчуються етапом змішування підрозділів із блоком даних. Цей додатковий крок робиться як захист від криптоаналізу. Щоб зашифрувати блок даних у Rijndael, спочатку потрібно виконати крок Add Round Key (XORing підрозділу з блоком) самостійно, потім регулярні раунди трансформації, а потім остаточний раунд із кроком змішування стовпця. Сам шифр визначається наступними кроками:

- початкове додавання раундового ключа;
- Nr-1 Раунди;
- заключний раунд.

Де Nr - кількість раундів, які необхідно виконати. Nr залежить від довжини блоку даних (Nb) та довжини ключа (Nk). Не враховуючи додатковий раунд, виконаний в кінці шифрування, кількість раундів у Rijndael становить: 9, якщо і блок, і ключ мають 128 біт, 11 якщо або блок, або ключ довжиною 192 біта, і жоден з них довше цього і 13, якщо або блок, або ключ, довжина 256 біт.

Раундове перетворення розбивається на шари. Ці шари є лінійним змішувальним шаром, який забезпечує високу дифузію протягом декількох раундів. Нелінійний шар, який в основному є додатком S-box Rijndael. І ключовий шар додавання, який є просто ексклюзивним або раунловим ключем і проміжним станом. Кожен шар розроблений таким чином, щоб мати свою чітко визначену функцію, яка підвищує стійкість до лінійного та диференціального криптоаналізу [8].

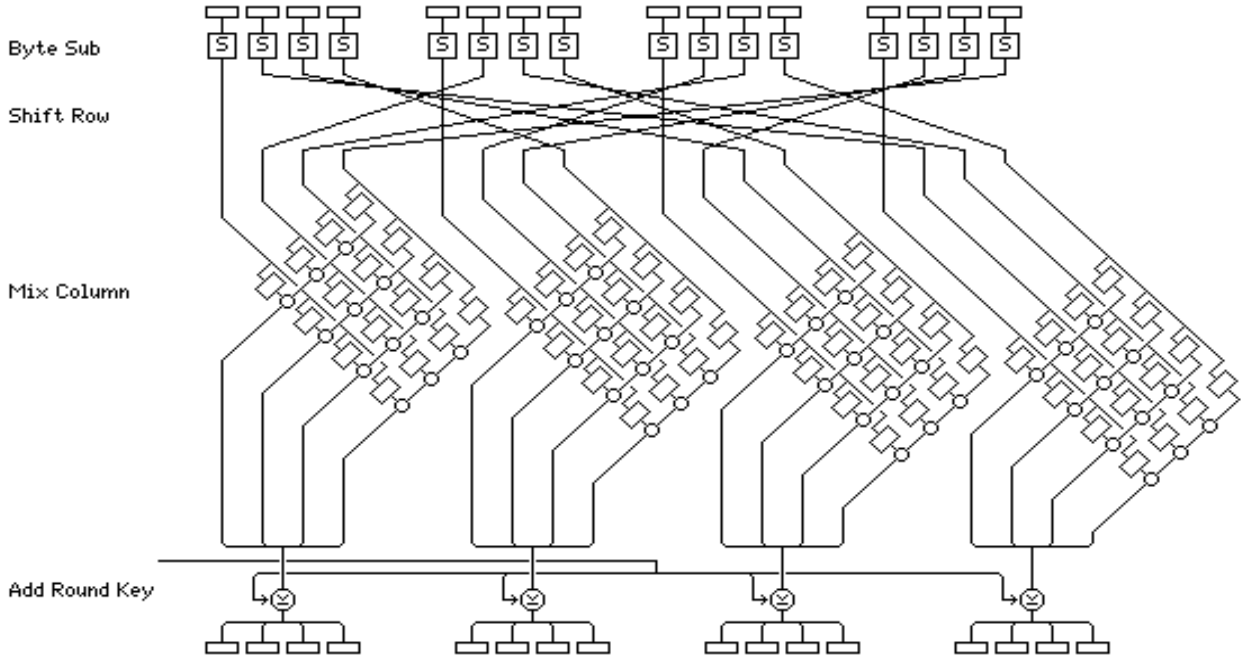

Рис. 4 процес Rijndael.

Ці шари здійснюються чотирма етапами трансформації. Крок "Байт" - це нелінійна підміна байтів. Перетворення рядка Shift - це циклічний зсув. Далі слідує етап MixColumn, в якому стовпці стану розглядаються як поліноми над кінцевим полем і по модулю множать на фіксований многочлен. Нарешті виконується крок Add Round Key(Рис. 4).

Крок "Байт" - це нелінійна заміна байтів, яка діє на кожен байт "стану" незалежно, де стан є проміжним результатом шифру. Операція цього кроку виконується за допомогою таблиці заміщення (S-box). ByteSub повертає слово, у якому кожен байт із вхідного стану відображається через S-поле у відповідні

байти. S-box Rijndael простий і складається з двох перетворень. Спочатку мультипликативну обернену інформацію беруть у кінцевому просторі поля GF, а потім застосовують афінне перетворення.

Дифузія визначається як мінімальна кількість активних S-таблиць у лінійній або диференціальній характеристиці . Дифузія важлива, оскільки в рамках криптоаналізу блочного шифру майже всі атаки мають складність, що залежить від кількості активних S-таблиць. На складність криптоаналітиків також впливає співвідношення введення / виводу окремих S-таблиць.

Лінійний дифузійний шар призводить до появи нижчих меж кількості активних S-таблиць. Вони мають виправдані нижні межі для нелінійного порядку, різниці лінійних функцій та стійкості до лінійного та диференціального криптоаналізу.

Крок Shift Row забезпечує, що різні байти кожного рядка не взаємодіють лише з відповідним байтом в інших рядках. Рядки держави циклічно зміщуються з різними зрушеннями. Рядок 0 не зміщений, рядок 1 переміщується над C1 байтами, рядок 2 над байтами C2 і рядок 3 над байтами C3. Зсуви зміщення C1, C2 і C3 залежать від довжини блоку Nb .

Крок Mix Column- це те, коли різні байти взаємодіють один з одним. Це змушує кожен байт у стовпці впливати на кожен інший байт.

Стан розглядається як поліноми, і перетворення складається з матричного множення стану з поліномом множення на поле фінітату. Етап трансформації стовпців-міксів - це єдине місце в раундовому перетворенні Rijndael, що колони змішуються. Цей крок працює з кроком Shift Row, щоб гарантувати, що всі частини блоку стикаються одна з одною.

Крок Add Round Key генерує новий раундовий ключ для наступного раунду перетворень.

2.2. Технологія «смарт-карт»

Смарт-карти зараз поширені в будь-якій економіці у всьому світі - ними користуються майже всі, хто отримує зарплату. Ідея смарт-картки полягає в тому, щоб зменшити архаїчний спосіб, коли люди переносять готівку та відчувають себе невпевнено завдяки чужим очам та численним випадкам посягань, пов’язаних із переміщенням грошей.

По суті, смарт-картки - це більш безпечний засіб здійснення фінансових операцій, полегшуючи життя у світі, який проходить безготівково. Їх називають смарт-картками, оскільки це не просто картки, це картки, що містять значну кількість приватної інформації та даних, обмежених певним користувачем: власником карти. Смарт-карти схожі на звичайні картки, але вони мають чіп пам'яті, що міститься у контактній панелі.

Розумні карти працюють не поодинці - для їх функціонування потрібен зчитувач смарт-карт. Коли певна відповідна особиста інформація була записана на смарт-карту її емітентами, вам знадобиться картридер, на якому ви зможете «прорізати» картку, пробити свій особистий код безпеки та здійснити будь-яку транзакцію.

Контактна накладка реально контактує з зчитувачем карт та згодом встановлює електронну взаємодію між собою та зчитувачем карт.

Потім це дозволяє здійснювати платежі через систему продажу (POS) або інші засоби масової інформації для здійснення транзакцій. Однак для деяких смарт-карт не потрібен інтерфейс для зчитування карт - безконтактні картки збільшують свою популярність завдяки своїй зручності.

Головною родзинкою технології смарт-карт є безпека, яку вона забезпечує користувачам. За допомогою смарт-картки користувачі можуть зберігати особисту інформацію, як-от банківські записи, посвідчення студентів для доступу до ексклюзивних бібліотек, ідентифікаційних карт компанії, щоб отримати доступ через комп'ютеризовані контрольно-пропускні

пункти безпеки, зберігання телефонних контактів, як на SIM-картах та багато інших величезних переваг, що гарантують безпеку особистої інформації дані.

Смарт-карти визначаються відповідно до:

1) Як читаються та записуються дані картки.

2) Тип мікросхеми, імплантованої всередині карти, та її можливості.

Побудова смарт-картки передбачає чотири основні етапи: проектування, виготовлення, кодування та завантаження даних:

- Проектування - це перший крок, який вимагає від дизайнера чи програміста виділити розмір пам’яті чіпу, вказати тактову частоту, типи пам’яті та операційну систему. Він також вимагає від програміста створити прикладне програмне забезпечення для картки, вказавши тип картки та будь-які інші функції, які він може включити до картки.
- Виготовлення чіпа - на цьому другому етапі в карту закріплюється чіп з кремнію. Ця кремнієва мікросхема з'єднується з роз'ємами з з'єднувальними проводами або їх пайкою, або з'єднанням між собою. Після цього мікросхема на підкладці дошки герметизується епоксидною смолою і приклеюється безпосередньо до підкладки картки. Підкладкою тут є пластик, який може бути виготовлений з полівінілхлориду (ПВХ) або будь-якого іншого синтетичного пластику.
- Кодування - це невід'ємний крок, на якому закладено основи функціональності картки. На цьому кроці коди вводяться в пам'ять чіпа за допомогою спеціальних команд.
- Завантаження даних - це етап, коли призначені особисті дані користувача завантажуються в мікросхему пам'яті.

Переважно всі чіпові карти побудовані з шарів різних матеріалів або підкладок, що при правильному зібранні надає картці певний термін експлуатації та функціональності. Типова сьогодні карта зроблена з ПВХ, поліестеру або полікарбонату. Шари картки спочатку друкуються, а потім ламінуються великим пресом. Наступним етапом у будівництві є розкрій або штампування. Далі слід вставити чіп і потім додати дані до картки. Загалом, може бути до 30 кроків у створенні картки. Загальна кількість компонентів, включаючи програмне забезпечення та пластмаси, може становити до 12 окремих елементів; все це в уніфікованому пакеті, який видається користувачеві як простий пристрій [9].

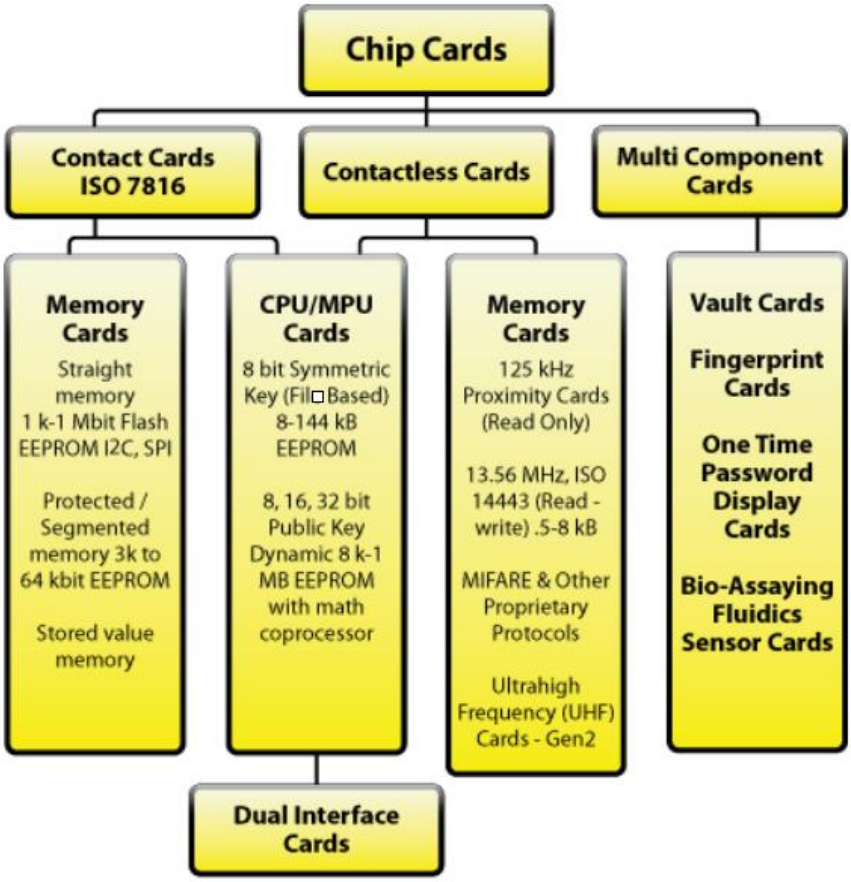

Рис. 5 види смарт-карт.

Контактні картки - це найпоширеніший тип смарт-карт. Електричні контакти, розташовані на зовнішній стороні карти, підключаються до картридера, коли карта вставлена. Цей роз'єм приєднаний до інкапсульованого чіпа на картці (Рис. 6).

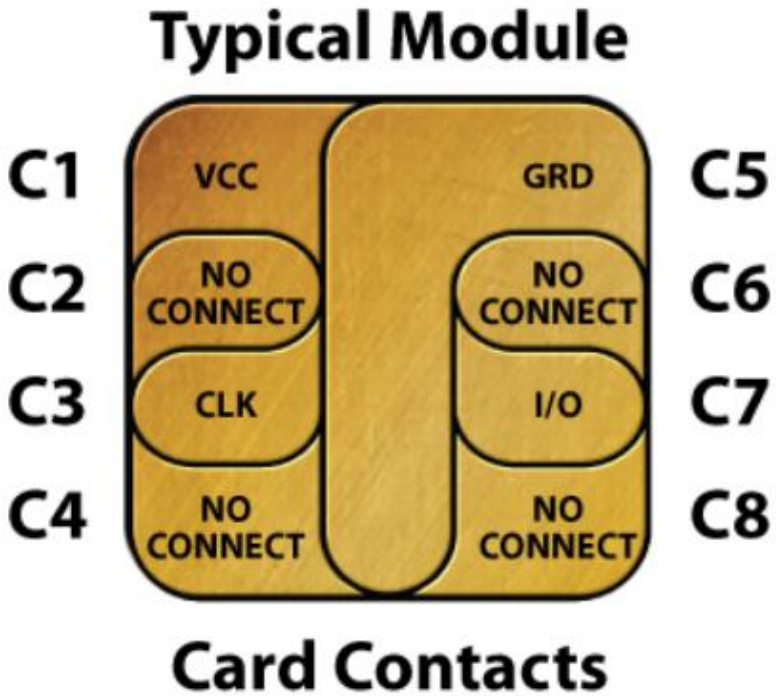

Рис. 6 чіп контактної смарт-карти

Підвищений рівень процесорної потужності, гнучкості та пам'яті призведе до збільшення витрат. Однофункціональні картки зазвичай є найбільш економічним рішенням. Необхідно вибирати правильний тип смарт-карти для своєї програми, визначивши необхідний рівень безпеки та оцінюючи вартість у порівнянні з вартістю інших апаратних елементів, знайдених у типовому робочому процесі. Усі ці змінні повинні бути зважені в залежності від очікуваного життєвого циклу картки. У середньому картки складають лише 10–15 відсотків загальної вартості системи, а інші 85 відсотків - інфраструктура, випуск програм, програмне забезпечення, читачі, навчання та реклама(Рис. 7).

Рис. 7 компромісні карткові функції

Карти пам'яті не можуть керувати файлами і не мають потужності для обробки даних. Всі карти пам'яті спілкуються з зчитувачами за допомогою синхронних протоколів. На всіх картах пам’яті ви читаєте та записуєте за фіксованою адресою на картці. Існує три основні типи карт пам'яті: пряма , захищена та збережена . Перш ніж проектувати ці картки в запропоновану систему, емітент повинен перевірити, чи підтримують читачі та / або термінали протоколи зв'язку мікросхеми. Більшість безконтактних карт є варіантами захищеної ідіоми карти пам'яті / сегментованої

Карти пам’яті:

- Прямі карти пам'яті - ці картки просто зберігають дані і не мають можливості обробки даних. Ці карти, часто виготовлені за допомогою напівпровідників I2C або серійних флеш-пам'яток, традиційно були найнижчою ціною за біт для пам'яті користувача. Зараз це змінилося, коли більша кількість процесорів будується для ринку GSM. Це різко зросло на користь цих пристроїв. Їх слід розглядати як дискети різного розміру без механізму блокування. Ці картки не можуть ідентифікувати себе читачеві, тому ваша хост-система повинна знати, який тип картки вставляється в зчитувач. Ці картки легко копіюються і не можуть бути відстежені за допомогою карткових ідентифікаторів.
- Карти пам'яті, захищені / сегментовані - Ці карти мають вбудовану логіку для управління доступом до пам'яті карти. Іноді їх називають картками інтелектуальної пам’яті, ці пристрої можна встановити для захисту деяких або всього масиву пам'яті. Деякі з цих карток можуть бути налаштовані для обмеження доступу як до читання, так і до письма. Зазвичай це робиться через пароль або системний ключ. Сегментовані картки пам'яті можна розділити на логічні розділи для запланованої багатофункціональності. Ці картки не дублюються легко, але, можливо, вони можуть бути представлені хакерами. Зазвичай їх можна відстежувати за допомогою ідентифікатора на картці.
- Картки пам'яті, що зберігаються - ці картки розроблені для конкретної мети зберігання вартості або жетонів. Карти є одноразовими або акумуляторними. Більшість карток такого типу містять постійні заходи безпеки на місці виготовлення. Ці заходи можуть включати ключі паролів та логіку, які жорстко закодовані виробником. Масиви пам'яті на цих пристроях налаштовуються як

декременти або лічильники. Ні для якої іншої функції не залишилося пам'яті або зовсім немає. Для простих додатків, таких як телефонна картка, чіп має 60 або 12 комірок пам'яті, по одному для кожного телефонного апарату. Очищення комірки пам’яті щоразу, коли використовується телефонний пристрій. Після використання всіх блоків пам'яті картка стає марною і викидається. Цей процес можна змінити у випадку перезарядних карт.

Мікропроцесорні багатофункціональні карти CPU / MPU - ці картки мають можливість динамічної обробки даних на карті. Багатофункціональні смарт-карти розподіляють пам'ять карт на незалежні розділи або файли, призначені для певної функції чи програми. Всередині карти знаходиться мікропроцесорний чи мікроконтролерний чіп, який управляє цим розподілом пам'яті та доступом до файлів. Цей тип мікросхем схожий з тими, які знаходяться у всіх персональних комп'ютерах, і при імплантації їх на смарт-карту керує даними в організованих файлових структурах за допомогою операційної системи карт. На відміну від інших операційних систем, це програмне забезпечення контролює доступ до пам'яті користувача на карті. Ця можливість дозволяє різним і безліччю функцій та / або різних додатків розміщуватися на картці, дозволяючи підприємствам видавати та підтримувати різноманітність «продуктів» через карту. Одним із прикладів цього є дебетова картка, яка також надає доступ до будівництва на кампусі коледжу. Багатофункціональні картки приносять користь емітентам, дозволяючи їм продавати свої товари та послуги за допомогою сучасних технологій транзакцій та шифрування. Зокрема, технологія забезпечує безпечну ідентифікацію користувачів та дозволяє оновлювати інформацію без заміни встановленої бази карт, спрощуючи зміни програми та зменшуючи витрати.

Для користувача картки багатофункціональність означає більшу зручність та безпеку, і, в кінцевому рахунку, консолідацію декількох карток до декількох вибраних, які служать багатьом цілям. Ця технологія забезпечує безпечну ідентифікацію користувачів та дозволяє оновлювати інформацію без заміни встановленої бази карт, спрощуючи зміни програми та зменшуючи витрати. Для користувача картки багатофункціональність означає більшу зручність та безпеку, і, в кінцевому рахунку, консолідацію декількох карток до декількох вибраних, які служать багатьом цілям. Ця технологія забезпечує безпечну ідентифікацію користувачів та дозволяє оновлювати інформацію без заміни встановленої бази карт, спрощуючи зміни програми та зменшуючи витрати. Для користувача картки багатофункціональність означає більшу зручність та безпеку, і, в кінцевому рахунку, консолідацію декількох карток до декількох вибраних, які служать багатьом цілям.

У цій категорії існує безліч конфігурацій мікросхем, включаючи мікросхеми, які підтримують функції криптографічної інфраструктури відкритих ключів (PKI) з вбудованими математичними копроцесорами або JavaCard з апаратними блоками віртуальної машини. Як правило, чим більше функцій, тим вище вартість.

Рис.8 смарт-карта MIFARE

Безконтактні картки - це смарт-карти, які використовують радіочастоту (RFID) між карткою та зчитувачем без фізичного вставлення карти. Натомість, картка передається по зовнішній стороні читача і читається. Типи включають карти близькості, які реалізовані як технологія лише для читання для побудови доступу. Ці карти функціонують з дуже обмеженою пам'яттю і спілкуються на 125 МГц. Іншим типом обмеженої картки є UHF-карта Gen 2, яка працює на частоті від 860 МГц до 960 МГц (Рис. 8).

Справжні безконтактні картки для читання та запису були вперше використані в транспортних програмах для швидкого зменшення та перезавантаження цін на проїзд, коли їх нижча безпека не викликала проблем. Вони спілкуються на частоті 13,56 МГц і відповідають стандарту ISO 14443. Вони також набувають популярності в роздрібній зберігається вартості, оскільки можуть пришвидшити транзакції, не знижуючи доходів від обробки транзакцій (тобто Visa та MasterCard), на відміну від традиційних смарт-карт.

Гібридні карти: мають кілька чіпів в одній карті. Зазвичай вони приєднуються до кожного інтерфейсу окремо, наприклад, мікросхема MIFARE та антени з контактним чіпом 7816 на тій же карті.

Подвійна інтерфейсна карта: такі картки мають один чіп, що управляє комунікаційними інтерфейсами. Мікросхема може бути приєднана до вбудованої антени за допомогою жорсткого з'єднання, індуктивного способу або за допомогою гнучких механізмів удару.

Карти багатокомпонентні**:** типи карток призначені для конкретного ринкового рішення. Наприклад, є картки, на яких вбудований датчик відбитків пальців. Або одна компанія створила карту, яка генерує одноразовий пароль та відображає дані для використання з додатком для онлайн-банкінгу. Карти сейфів мають перезаписані магнітні смуги. Кожна з цих технологій є специфічною для конкретного постачальника і, як правило, запатентована.

Очікувана форма для карт часто називається CR80. Банківські та особисті картки регулюються специфікацією ISO 7810. Але ця форма - не єдиний форм-фактор, в якому розміщуються картки. У всьому світі використовуються вирізи карт з модулями та / або антенами. Найпоширеніші форми - SIM. Карти SD та MicroSD тепер можуть розгортатися з міцністю чіпів смарт-карт. Також доступні жетони флеш-накопичувача USB, які використовують ту саму технологію картки в іншому форм-факторі.

Два основні типи операційних систем смарт-карт - це (1) фіксована структура файлів та (2) динамічна система додатків . Як і для всіх типів смарт-карт, вибір операційної системи карт залежить від програми, для якої призначена карта. Інша визначальна різниця полягає у можливостях шифрування операційної системи та мікросхеми. Типи шифрування - це симетричний ключ та асиметричний ключ (відкритий ключ) .

Багато функцій, такі як конкретні алгоритми шифрування, включені до апаратних та програмних бібліотек архітектури чіпів. Це часто може

призвести до того, що виробник карт не підтверджує свою конструкцію, якщо їх операційні системи карт переносяться лише на певний пристрій. Інструменти та проміжне програмне забезпечення, що підтримують операційні системи карт, настільки ж важливі, як і сам чіп. Інструменти для реалізації проекту повинні бути простими у використанні та давати можливість швидко розгортати проект [10].

Операційні системи карт:

- Операційна система картки з виправленою структурою файлів - цей тип розглядає карту як захищений обчислювальний і запам'ятовуючий пристрій. Файли та дозволи встановлюються заздалегідь емітентом. Ці конкретні параметри є ідеальними та економічними для фіксованого типу структури та функцій карт, які не зміняться найближчим часом. Багато типів захищеної цінності та охорони здоров’я використовують цей тип карт. Прикладом такого виду картки є багатофункціональний значок або обліковий запис недорогих службовців. На відміну від деяких упереджених статей, ці картки стилів можна використовувати дуже ефективно із збереженим біометричним компонентом та зчитувачем. В усьому світі ці типи мікропроцесорних кар т є найпоширенішими.
- Операційна система динамічної програми - цей тип операційної системи, що включає в себе JavaCard® та фірмові різновиди карт MULTOS, дозволяє розробникам безпечно створювати, тестувати та розгортати різні програми на картах. Оскільки операційні системи та програми карт є більш окремими, можна проводити оновлення. Приклад картки - це SIM-карта для мобільного GSM, де оновлення та безпеку завантажуються на телефон і динамічно змінюються. Цей тип розгортання картки передбачає, що програми в цій галузі змінюватимуться за дуже короткий

проміжок часу, що обумовлює необхідність динамічного розширення карти як обчислювальної платформи. Витрати на зміну додатків у цій галузі високі, що зумовлено екосистемними вимогами безпеки для обміну ключами з кожною обліковою інформацією. Це змінна, яку слід ретельно вивчити на етапі проектування карткової системи.

Невід'ємною частиною технології смарт-карти є зчитувачі та термінали. Зчитувачі та термінали працюють із смарт-картками для отримання інформації про карту та здійснення транзакції. Як правило, зчитувач взаємодіє з ПК для його обробки. Термінал - це автономний пристрій обробки. І зчитувачі, і термінали читають і записують певні данні на смарт-карти. Зчитувачі в основному поділяються на 2 види : перший для контактних смарт-карт та другий для безконтактних.

Контактний зчитувач – цей тип зчитувача вимагає фізичного підключення до карт, здійсненого шляхом вставлення карти в зчитувач. Це найпоширеніший тип зчитувача для таких програм, як ідентифікатор та збережене значення. Комунікація карт-читач часто є лише ISO 7816 T = 0. Це спілкування має перевагу прямого зв'язку з читачем і вважається більш безпечним. Інша перевага - швидкість. Цей інтерфейс дозволяє переносити більші дані без накладних проблем зіткнення та бездротового злому, які виникають в результаті переміщення карти в діапазон антени зчитувача.

Рис. 9 зчитувач для безконтактних смарт-карт.

Безконтактний зчитувач - цей тип зчитувача працює з радіочастотою, яка повідомляє, коли карта наближається до зчитувача. Багато безконтактних читачів розроблені спеціально для програм оплати, фізичного контролю доступу та транспорту. Домінуючим протоколом згідно ISO 14443 є MIFARE, за яким слідують стандарти EMV. Зображен зчитувач для безконтактних смарт-карт, саме такий ми будемо використовувати при розроці нашого додатку (Рис. 9).

Термінали - на відміну від читачів, термінали більше схожі на автономний ПК, з більшістю представлених операційних систем та інструментів розробки. Термінали часто є специфічними для випадків використання, таких як безпека, охорона здоров’я та інформаційна служба (точка продажу).

Підключення в терміналах зазвичай здійснюється через протокол управління передачею / протокол Інтернету (TCP-IP) або мережу GSM. На сьогоднішній день у багатьох терміналах є звичайна операційна система, яка спрощує розгортання, наприклад, Datastrip з Windows CE або Exadigm з Linux [9].

При розробці нашого додатку ми використали безконтактну смарт-карту MIFARE 2K та зчитувач для безконтактних смарт-карт ACR 1222L.

2.3 Програмна платформа.

Для реалізації додатку в якості платформи було обрано саме клієнтський додаток для Windows, оскільки станом на 2020 рік Microsoft офіційно підтвердив, що тільки ОС Windows 10 встановлено на один мільярд персональних комп'ютерів, згідно інформації компанії про кількість активних пристроїв на березень цього року. Це робить ОС Windows однією з найпопулярніших ОС у світі[12].

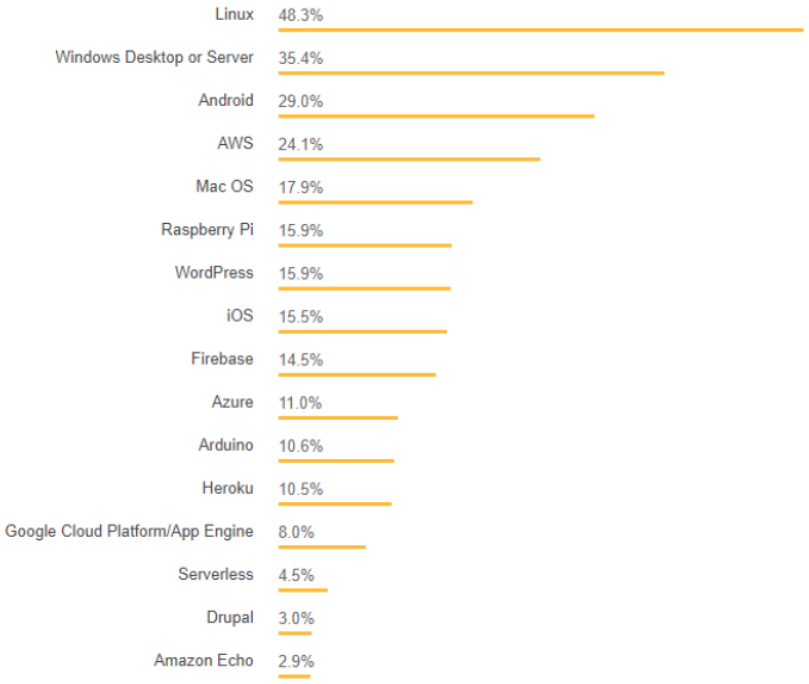

Рис.10 Результат опитування Stack Overflow 2018 щодо платформи.

Результат опитування респодентів Stack Overflow 2018 показав, що вони розробляли програми здебільшого для Linux-систем, а також для десктопних та серверних версії Windows (Рис.10).

Для реалізації програмної частини була використана мова програмування C# та .NET Framework, що дозволило створити простий та надійний додаток.

C# - проста, сучасна об'єктно-орієнтована мова прогармування, яка підтримує сучасний функціонал для всіх видів розробки програмного забезпечення.

Ця мова дозволяє розробникам створювати прості та надійні програми, які працюють на .NET Framework або .NET Core. C# використовується для створення клієнтських додатків Windows, MacOS, Linux, розподілених компонентів, веб-служб, клієнт-серверних додатків, додатків баз даних і інших програмних продуктів. Visual C# надає доступний редактор коду, відладчик та багато інструментів, які полегшують розробку додатків.

Синтаксис C # простий і легкий в освоєнні.Для тих хто вже знайомий з C, C ++ або Java, не буде складністю почати програмувати на C#. Ця мова програмування має такий же синтаксис фігурної дужки як у C, C++ та Java. C# має такий синтаксис, порівняно з C ++, який може спрощувати багато складнощів і надає потужні функції: типи значень з нулями, делегати, прямий доступ до пам'яті, перерахування. C# може підтримувати загальні методи, які забезпечують підвищену безпеку. Важливо те, що з новим ставленням Microsoft до відкритого коду можна бути впевненим, що C# має великі перспективи для подальшого розвитку.

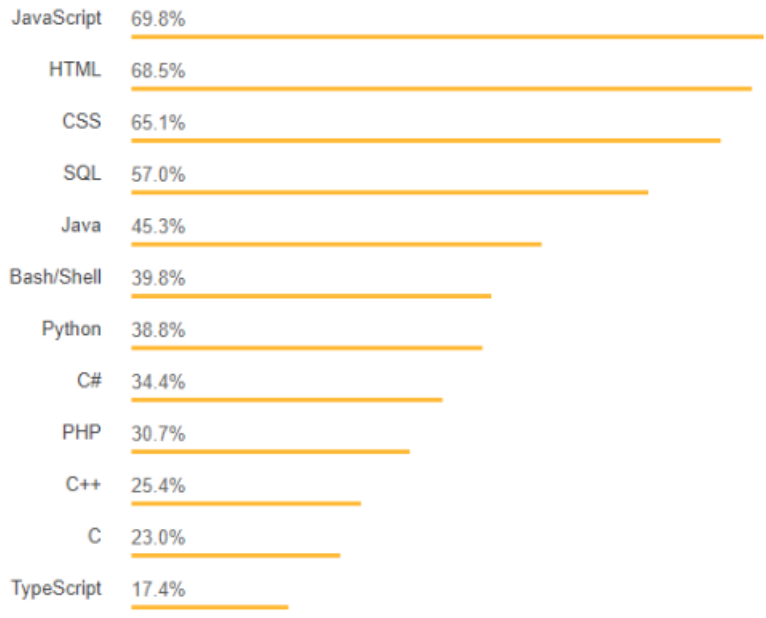

Рис. 11 результат опитування Stack Overflow 2018 щодо мови програмування

У опитуванні розробників StackOverflow у 2018 році C# посів 8 місце у списку найпопулярніших технологій та його мають у своєму арсеналі 34.4% (Рис.11). І з кожним роком прихильників цієї мови програмування становиться більше[11].

.NET Framework - це програмне забезпечення, розроблене Microsoft, яка працює в основному на ОС Windows . Він включає велику бібліотеку класів під назвою Framework Class Library (FCL) і забезпечує інтероперабельність мови (кожна мова може використовувати код, написаний іншими мовами) для кількох мов програмування. Програми, написані для .NET Framework, виконуються в програмному середовищі під назвою Common Language Runtime (CLR). CLR - це віртуальна машина додатків, який надає такі послуги, як безпека, управління пам'яттю та обробка виключень. FCL і CLR разом складають .NET Framework (Рис. 12). FCL надає інтерфейс для користувача , доступ до даних, підключення до баз даних, криптографічні функції, розробка веб-додатків, цифрові алгоритми і мережеві комунікації. Інтегроване середовище розробки для програмного забезпечення .NET - Visual Studio .

Рис.12 стек компонентів .NET Framework

3. ДОДАТОК

3.1. Інструкція для використання

Розроблений додаток ми отримали дуже простим для зрозуміння і використання. При відкритті додатку користувач побачить меню для сдійснення шифрування та дешифрування файлів (Рис. 13)

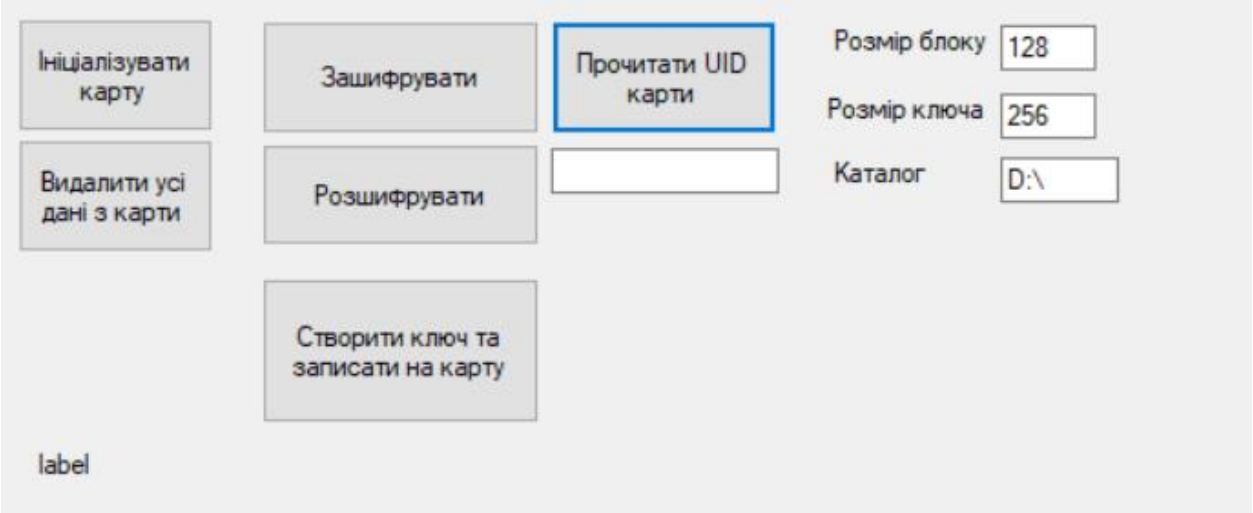

Рис. 13 контекстне меню.

Перед початком використання додатку користувач повинен підключити зчитувач для безконтактних смарт-карт(у нашому випадку ACR 1222L) та прикласти смар-карту(у нашому випадку MIFARE 2K) до цього зчитувача. Слід зауважити, що під час усіх процесів смарт-карта повинна знаходитись на зчитувачі, щоб уникнути помилок так втрати даних.

Розглянемо пункти меню, за допомогою яких ми можемо підготувати смарт-карту для використання:

- «Видалити усі дані з карти» - ящо смарт-карта раніше використовувалась, ми стираємо усі дані, які на ній раніше зберігались.
- «Ініціалізувати карту» - надати певний ID та поставити певний криптографічний ключ для доступу до цієї смарт-карти.

- «Почитати UID карти» - перевірка чи пройшла карта ініціалізацію, якщо так то у пустому рядку з’явиться UID карти.

На даний момент ми підготвували нашу смарт-карту для подальшого її використання.

Розглянемо пункти меню, за допомогою яких ми можемо зашифрувати наші файли:

- «Створити ключ та записати на карту» - генерується наш криптографічний ключ і записується на смарт-карту.
- На цьому етапі користувач повинен встановити розмір ключа(128/192/256 біт) та розмір блоку(128 біт) для алгоритму Rijndael та вписати каталог де буде здійснюватись пошук файлу, який потрібно зашифрувати.
- «Заштфрувати» - при натисканні на цю кнопку відкритється діалогове вікно файлів, у тому каталозі, який ми указали у попередньому пункті (Рис. 14)

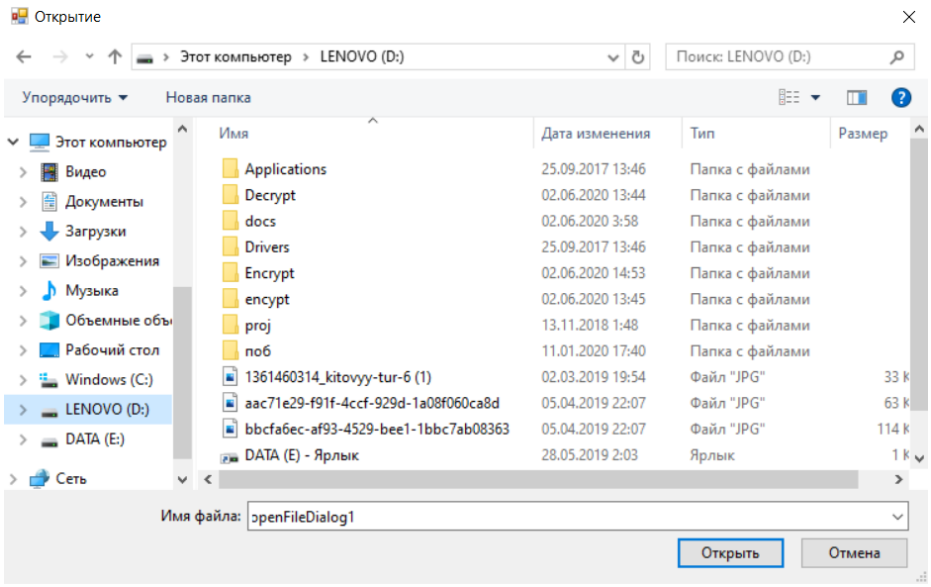

Рис. 14 діалогове вікно файлів.

Можна переглянути текстовий файл до шифрування (Рис. 15) та файл після шифрування(Рис. 16).

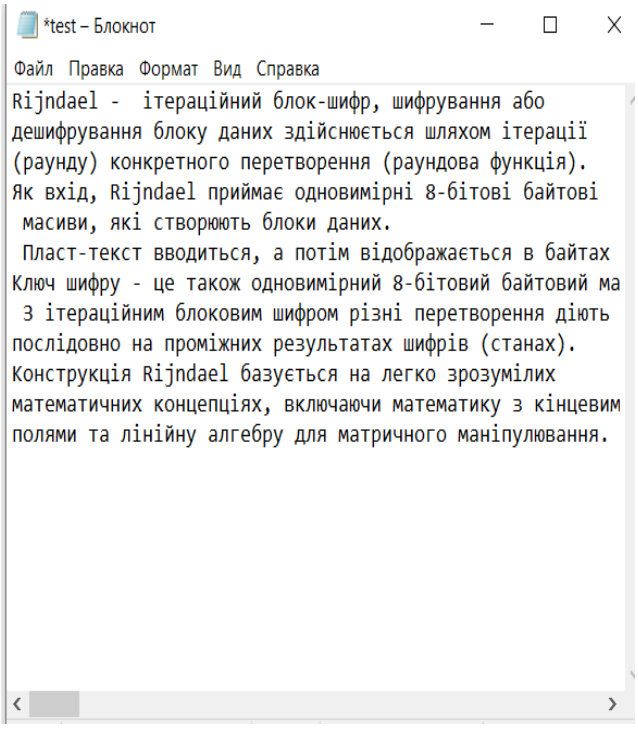

Рис. 15 файл .txt до шифрування.

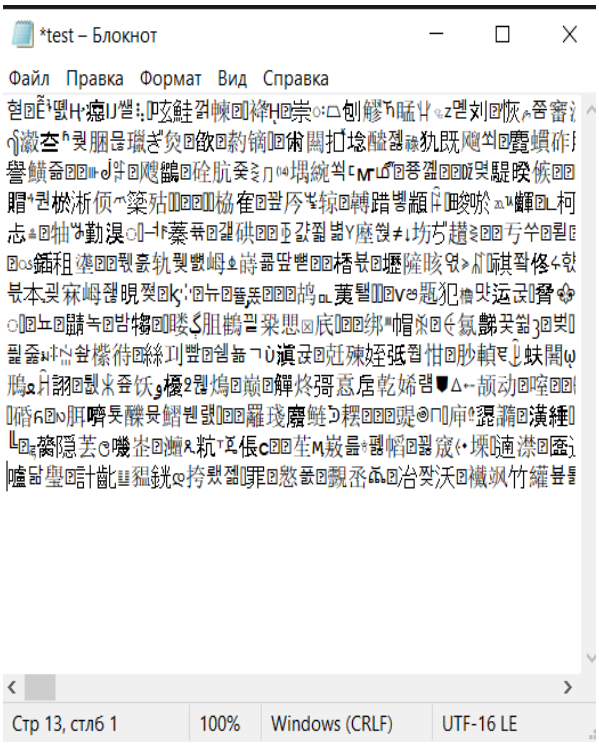

Рис. 16 файл .txt після шифрування

Для прикладу було використано саме файл розширення .txt, тому що файли з цим розширення відкриваються, при шифруванні файлів розширення .png, .docx та .pdf ці файли не відкриваються, а видає помилку що файли пошкоджені. Хочеться зазначити що великою перевагою є те що файл після шифрування залишається в тому розширенні в якому і був на самому початку. Поки ви не відкриєте файл, ви не зрозумієте зашифрован він чи ні.

Для того щоб розшифрувати файл, користувач повинен прикласти смарт-карту до зчитувача та натиснути «Розшифрувати» - на цьому етапі здійснюється зчитування з карти та відкривається діалогове вікно файлів(Рис. 14). Потрібно вибрати файл який був зашифрован. Файл розшифрован (Рис. 17)

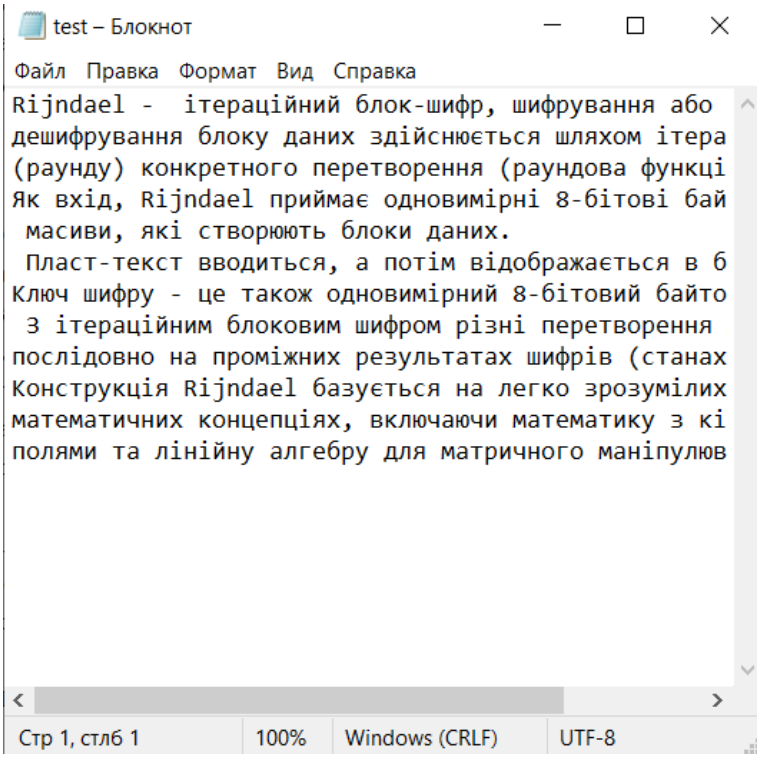

Рис. 17 файл .txt після розшифрування.

Файли с іншими розширеннями .png, .docx та .pdf також успішно проходять етап розшифрування.

3.2 Опис модулів

Для забезпечення фукнціонування додатку були розроблені і використані наступні модулі:

- Модуль зчитувача відповідає за передачу інформації з та на смарт-карту.
- Модуль смарт-карти дозволяє звертатись до її пам’яті та вести запис інформації на цю карту і зчитувати певні дані з неї.
- Модуль ініціалізації смарт-карти надає користовачу змогу надати цій карті певни ID та зашифрувати її криптографічним ключем для запобігання зчитування цієї карт іншим користувачем.
- Модуль зтирання даних дає змогу користувачу повність зтерти усі дані з карти.
- Модуль UID карти дає впевнитись користувачу, що карта пройшла етап ініціалізації та має захист.
- Модуль генерації ключа відповідає за генерацію криптографічного ключа та запису на смарт-карту.
- Модуль шифрування відповідає за кодування певного файлу алгоритмом Rijndael.
- Модуль розшифрування відповідає за декодування файлу, який був зашифрован раніше, використовуючи ключ, який зберігається на смарт-карті.

ВИСНОВОКИ

Створений додаток виконує усі функції, які були поставленні за мету на самому початку та має простоту у використанні,. Хоч він має багато можливостей для вдосконалення, але вже може бути самостійним продуктом, для тих кто хоче захистити певні файли на своєму комп'ютері від зловмисників.

Розроблений додаток покриває значну частину потреб користувачів і має велику перевагу у тому, що при шифруванні файлу не створюється новий з іншим розширенням, з цього можна сказати, що ніхто не зможе здогадатись де саме знаходиться файл і який саме файл зашифровано. Крім цього додаток надає декілька рівнів захисту: по-перше ключ зберігаєтсья тільки на фізичному носії – смарт-карті; по-друге навіть якщо зловмисник заволодіє вашою смарт-картою і якимось чином зашифрованим файлом, без інформації про смарт-карту та криптографічного ключа, який був встановлений на неї він не зможе прочитати дані з цієї карти.

Під час розробки додатку тестувались файли з різними розширеннями та розмірами (від 0.5Мб до 7Гб) усі вони успішно пройшли етапи шифрування та розшифрування, також тестувались усі можливі розміри ключей для алгоритму Rijndael і відхилень не спостерігалось .

При розробці спостерігались вектори, за якими додаток може бути вдосконалений:

- Кодування ішними алгоритмами окрім Rinjdael.
- Покращення захисту самої смарт-карти.
- Кодування не тільки файлів но і цілих каталогів.
- Розробити більш приємний інтерфейс для користувача.
- Реалізувати цей додаток для інших видів смарт-карт окрім безконтактних MIFARE 2K.

Розроблений додаток буде частково викладений у публічний доступ, що дозволить іншим покращити його та створити власні модифікації

Окрім плюсів у цьому способі захисту інформації є декілька недоліків : по-перше якщо ви втратите фізичний носій – смарт-карту, то вже не зможете відновити ващі дані, у вас залишиться зашифрований файл без змоги його відновити; по-друге цей додаток потребує значних витрат, для його використання вам знадобиться смарт-карта MIFARE 2K та зчитувач для безконтактних смарт-карт ACR 1222L.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Безпечна файлова система (SFS) для DOS / Windows. [Електроний ресурс].
   URL www.cs.auckland.ac.nz/~pgut001/sfs/index.html (дата звернення 25.04.2020).
2) Домашня сторінка програмного забезпечення Jetico, Inc. [Електроний ресурс]. URL www.jetico.com , 2002. (дата звернення 26.04.2020).
3) Корпорація Майкрософт. Шифрування файлової системи для Windows 2000. Технічний звіт, липень 1999 р. . [Електроний ресурс].
   URL Www.microsoft.com/windows2000/techinfo/howitworks/security/encrypt.asp (дата звернення 26.04.2020).
4) Охорона конфіденційності GNU [Електроний ресурс].
   URL www.gnupg.org (дата звернення 26.04.2020).
5) Офційна документація veracrypt [Електроний ресурс].
   URL https://archive.codeplex.com/?p=veracrypt (дата звернення 2.05.2020).
6) Ліцензія axcrypt [Електроний ресурс]. URL
   https://www.axcrypt.net/information/license/ (дата звернення 3.05.2020)
7) ЕЛЕКТРОННА БІБЛІОТЕКУ ДЕРЖАВНОГО УНІВЕРСИТЕТУ ТЕЛЕКОМУНІКАЦІЙ Б. Шнайер. *Прикладна криптографія* . John Wiley & Sons, 2 видання, жовтень 1995 року. [Електроний ресурс].
   URL http://www.dut.edu.ua/uploads/l_1134_27449793.pdf (дата звернення 21.05).
8) Технології захисту інформації [Електронний ресурс] : підручник для студ. спеціальності 122 «Комп’ютерні науки», спеціалізацій «Інформаційні технології моніторингу довкілля», «Геометричне моделювання в інформаційних системах» / Ю. А. Тарнавський; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 2,04 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 162 с.

9) Будова смарт-карт [Електроний ресурс]. URL https://www.cardlogix.com/ (дата звернення 21.05.2020).
10) Основи смарт карт [Електроний ресурс]. URL http://www.smartcardbasics.com/ (дата звернення 21.05.2020).
11) Результати щорічного опитування Stack Overflow 2018 [Електроний ресурс]. URL https://proglib.io/p/stack-overflow-2018 (дата звернення 24.05.2020).
12) Новини компанії Windows[Електроний ресурс]. URL https://blogs.windows.com/ (дата звернення 24.05.2020).